10-2020

From Inductive to Deductive Learning

Mikhail Mayers
Portland State University

Brian Henson
Portland State University

FROM INDUCTIVE TO DEDUCTIVE LEARNING

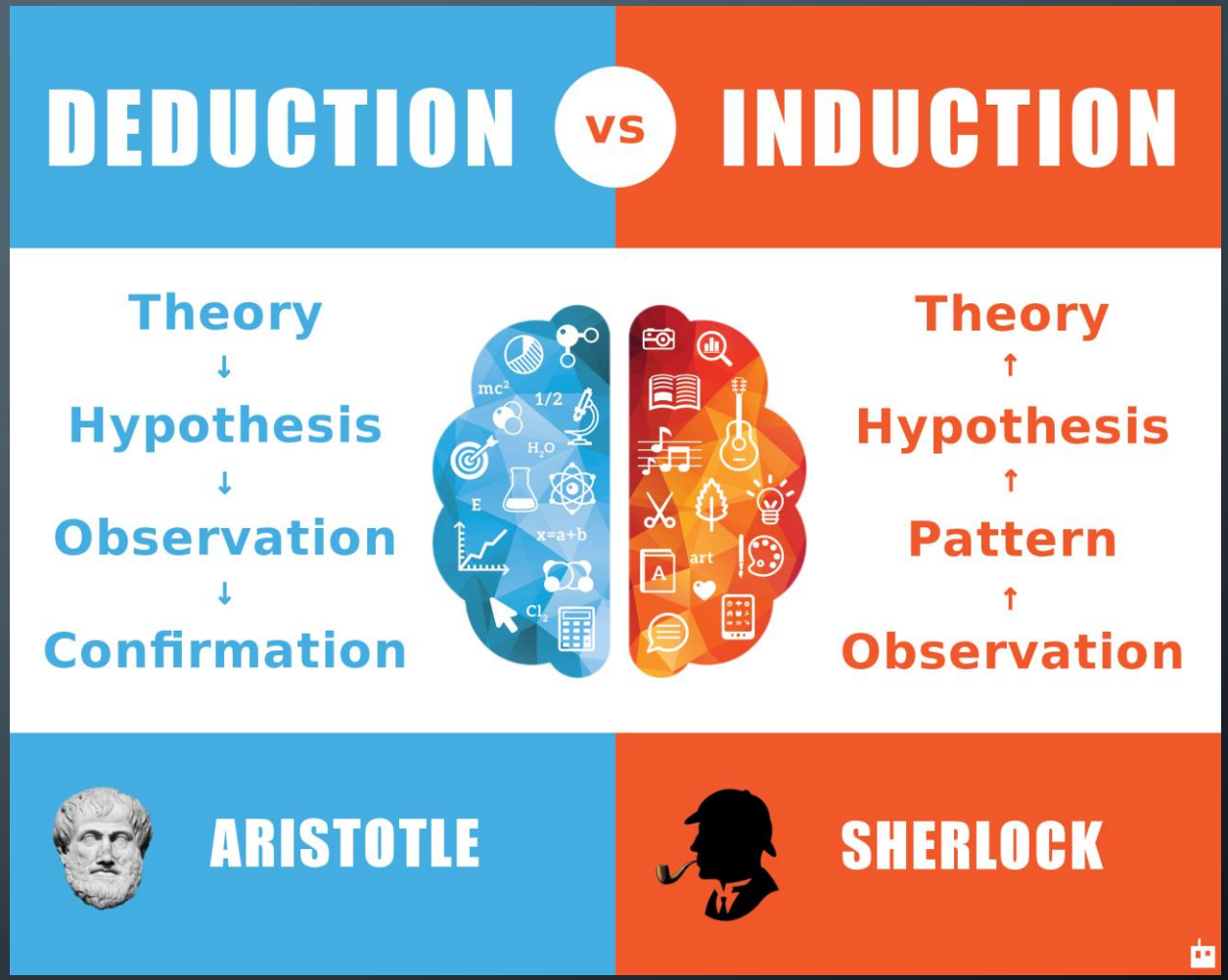
MIKHAIL MAYERS

START WITH DEFINITIONS

- Inductive reasoning
- Deductive reasoning
- Analogical reasoning

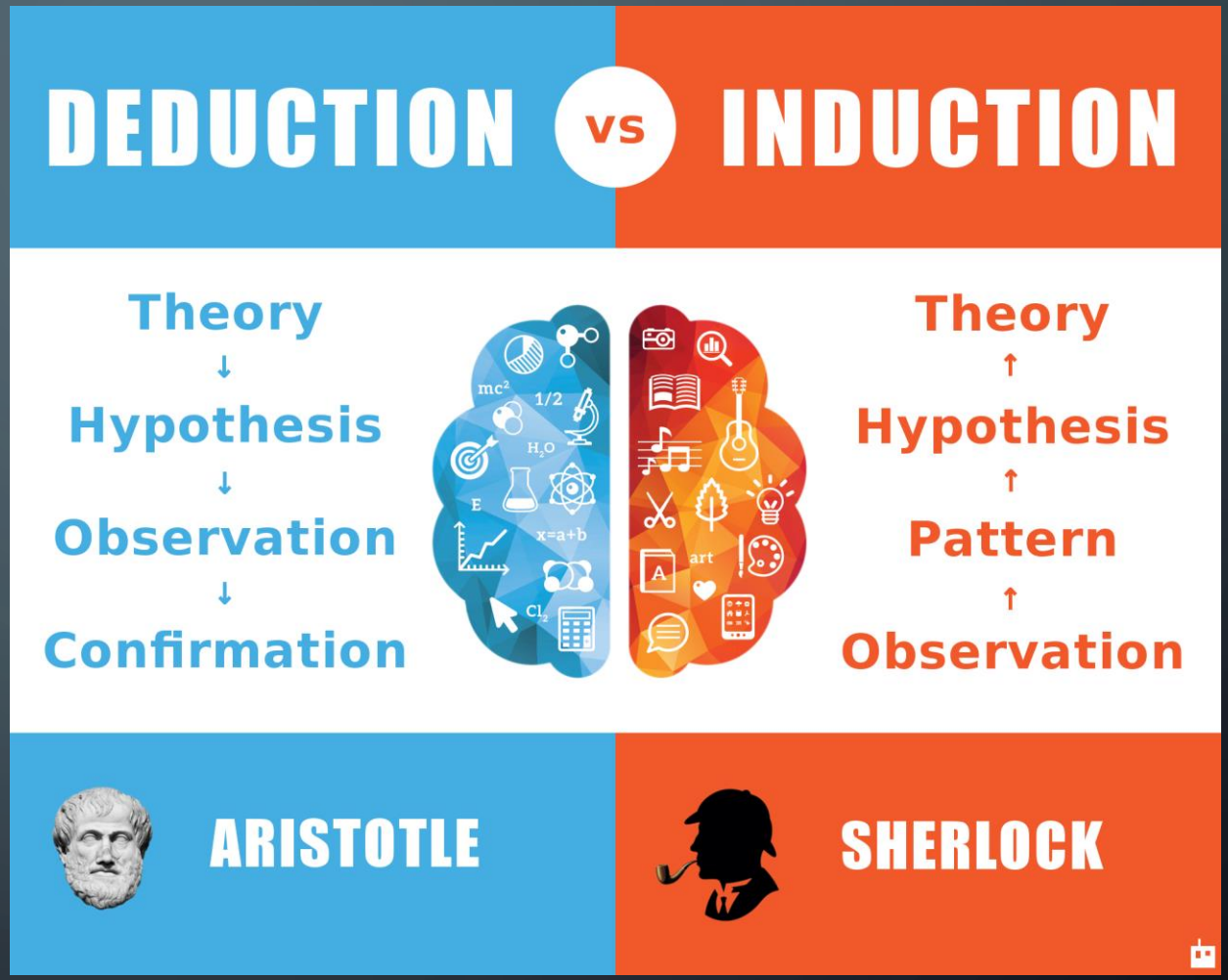
INDUCTIVE REASONING

- Deriving general principals from specific observations
- Ex: All organisms are made of cells based on years of findings



DEDUCTIVE REASONING

- Use what you know to be true in general to decide what must be true n a specific case
- Ex: If all organisms are made of cells and humans are organisms, then humans are made of cells



ANALOGICAL REASONING

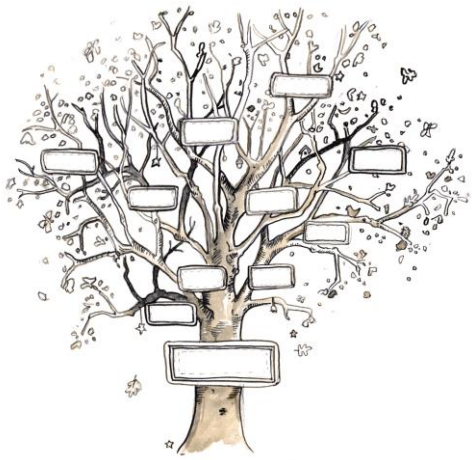
- TWO THINGS ARE SIMILAR, WHAT IS TRUE OF ONE IS ALSO TRUE OF THE OTHER

- Earth
  - Orbits sun
  - Has a moon
  - Subject to gravity
  - Contains water
  - Has life
- Mars
  - Orbits sun
  - Has a moon
  - Subject to gravity
  - Contains frozen water
  - May have life?

```
/* Facts */
male(jack).
male(oliver).
male(ali).
male(james).
male(simon).
male(harry).
female(helen).
female(sophie).
female(jess).
female(lily).

parent_of(jack,jess).
parent_of(jack,lily).
parent_of(helen, jess).
parent_of(helen, lily).
parent_of(oliver,james).
parent_of(sophie, james).
parent_of(jess, simon).
parent_of(ali, simon).
parent_of(lily, harry).
parent_of(james, harry).

/* Rules */
father_of(X,Y):- male(X),
    parent_of(X,Y).

mother_of(X,Y):- female(X),
    parent_of(X,Y).

grandfather_of(X,Y):- male(X),
    parent_of(X,Z),
    parent_of(Z,Y).

grandmother_of(X,Y):- female(X),
    parent_of(X,Z),
    parent_of(Z,Y).
```

PROLOG

- Classic example is a family tree
  - By only giving the facts and rules
  - Prolog will build relationships between the facts with given rules



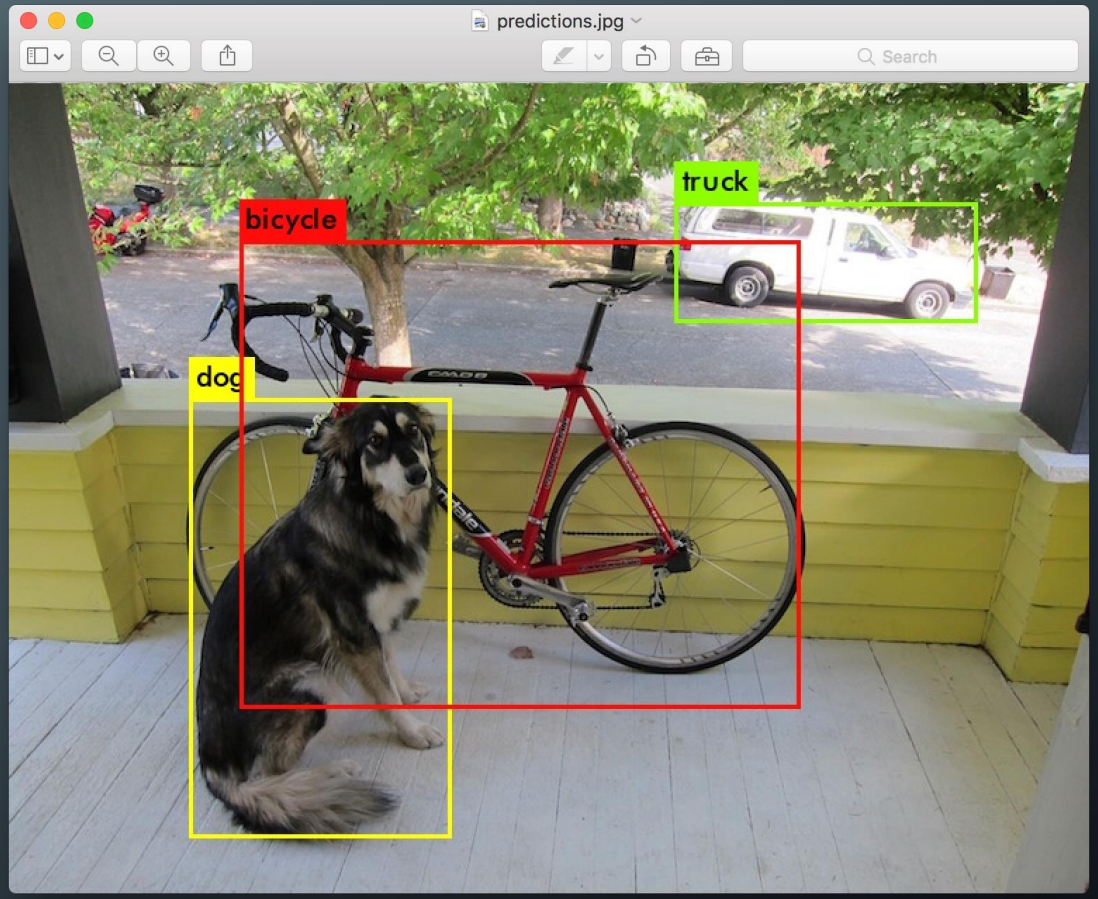
```
?- mother_of(X,jess).
X = helen .

?- parent_of(X,simon).
X = jess .

?- sister_of(X,lily).
X = jess .
```

PROLOG

- Relationships of family members are structured
- The data base can be queried
  - Prolog will return data

OBJECT DETECTION

- So much data information from objects
  - name
  - Position
  - Color



INFORMATION

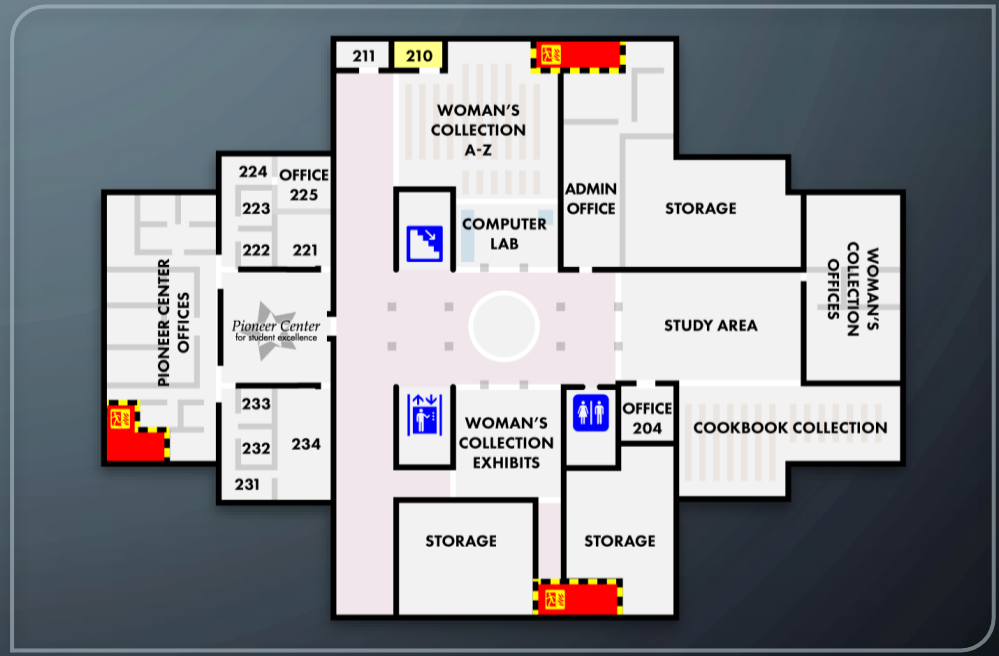
- Darknet
  - Conveniently draw bounding box
  - Using the coordinates used to draw the bounding box
    - Center Point (relative position)
    - Color(for most objects)
    - size
  - Name of the object
    - That gave give a type
      - Plant
      - Shapes
      - Animal

PURPOSE OF PROJECT

- Monkey and Banana
  - Robot must find object
  - Object is too high
  - Needs to stack boxes to reach object

BANANA IN A BOX

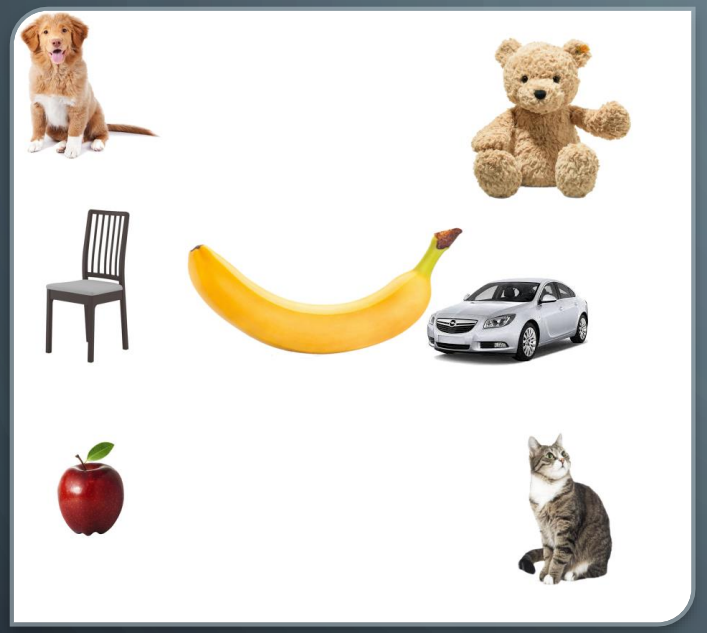
- Banana is inside a box
- Robot gets a hint of the location
- Narrows search space to find banana



GOAL

- Recognize an object
- Gather attributes from the objects
- Put the objects with their attributes into Prolog
- Use prolog to build rations between the objects

DATA FRAME



Name	X Center	Y Center	Color	Size
teddy0	170.0	51.5	Coffee	7569.0
banana	125.5	96.5	White	12168.0
dog0	30.0	39.5	Pale Gold	4484.0
chair0	33.5	97.0	Ash Grey	4316.0
car0	161.0	96.5	Liver	3696.0
apple0	31.0	146.5	Barn Red	1880.0
cat0	167.5	161.0	Pastel Brown	4488.0

- The attributes are put into a data frame
  - Values can become general facts
    - Ex *Teddy is above and to the right of the banana*
    - Ex *Banana is large than the apple*

RELATION AND QUERIES

The information can be added to prolog

(In relation to Banana)

```
right_of(banana,teddy0)
above_of(banana,teddy0)
smaller_than(banana,teddy0)
left_of(banana,dog0)
above_of(banana,dog0)
smaller_than(banana,dog0)
left_of(banana,chair0)
below_of(banana,chair0)
smaller_than(banana,chair0)
right_of(banana,car0)
above_of(banana,car0)
smaller_than(banana,car0)
left_of(banana,apple0)
below_of(banana,apple0)
smaller_than(banana,apple0)
right_of(banana,cat0)
below_of(banana,cat0)
smaller_than(banana,cat0)
```

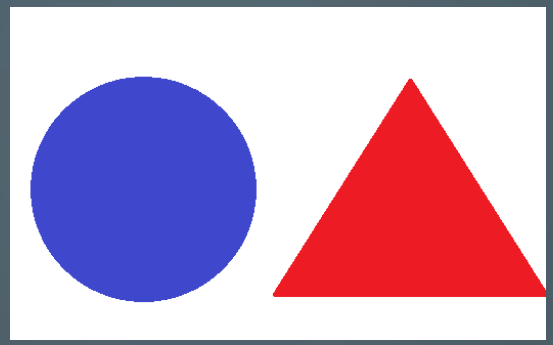
The information can be queried

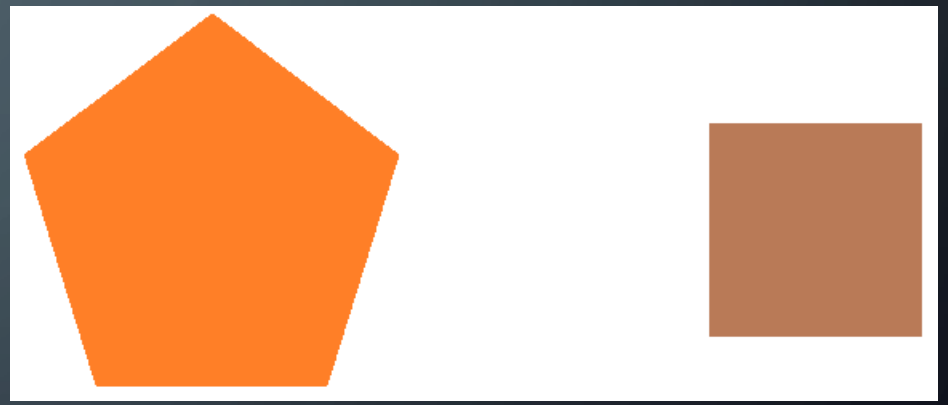
What's right of the Banana

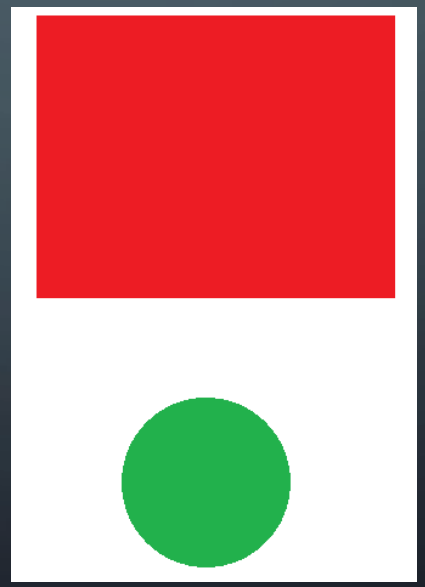
```
=====================================
Objects right of Banana
=====================================
teddy0
car0
cat0
=====================================
Objects Bigger than Banana
=====================================
nothing
```

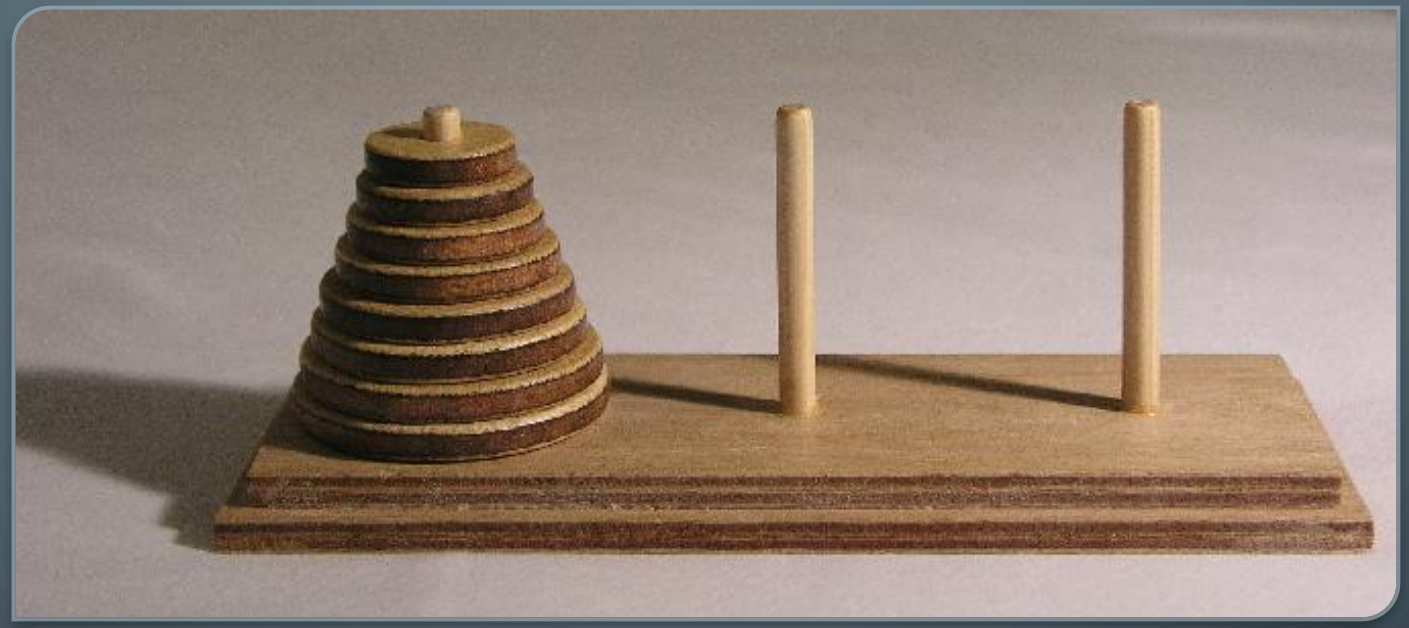
DETECTION

- Can check:
  - Color
  - Shape/object type
  - Above/below
  - Right/left
  - Inside/Outside





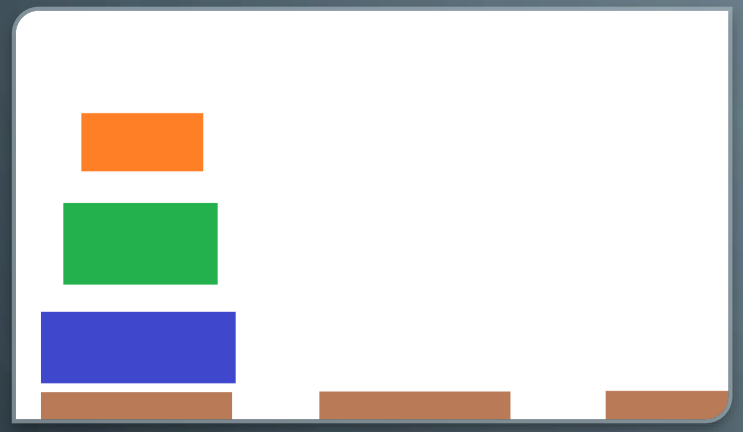




GAMES

TOWER OF HANOI

SAME IDEA

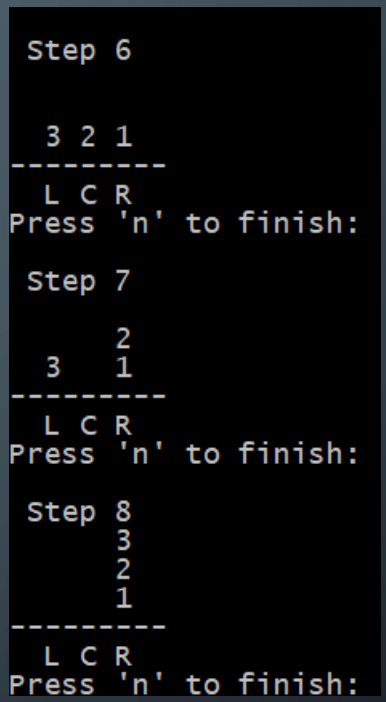


```
1 :147 :528
2 :146 :300
3 :151 :130
284
1 147
2 146
3 151
{'right': deque([]), 'center': deque([]), 'left': deque([1, 2, 3])}
{'right': deque([]), 'center': deque([]), 'left': deque([1, 2, 3])}

 Step 1
  3
  2
  1
---------
  L C R
```

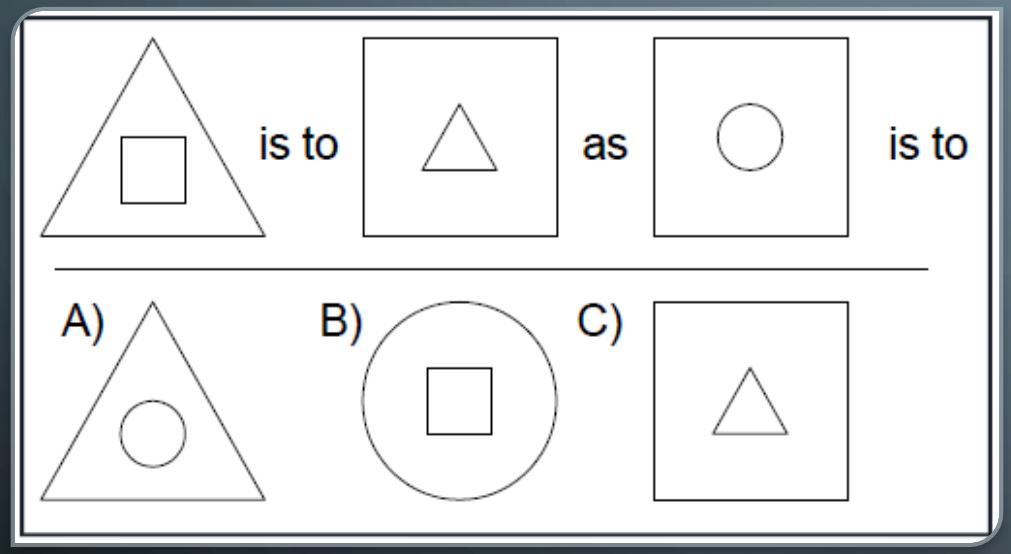
- Take in visual data
- Put it in a table
- Give that data to prolog

PROLOG SAVES THE DAY

Prolog solves the problem with its deductive reasoning.



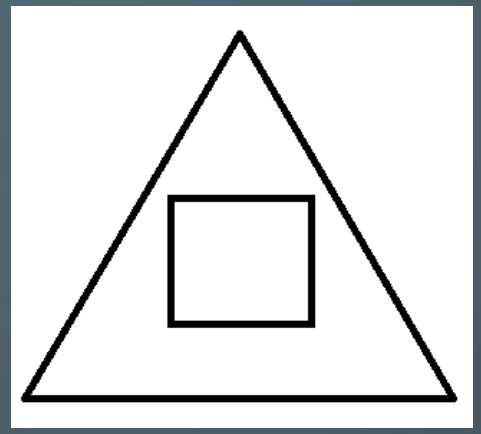
ANALOGY

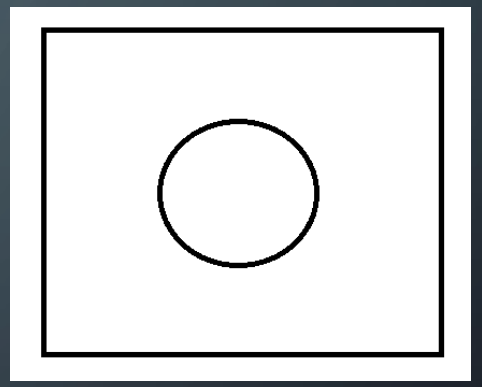


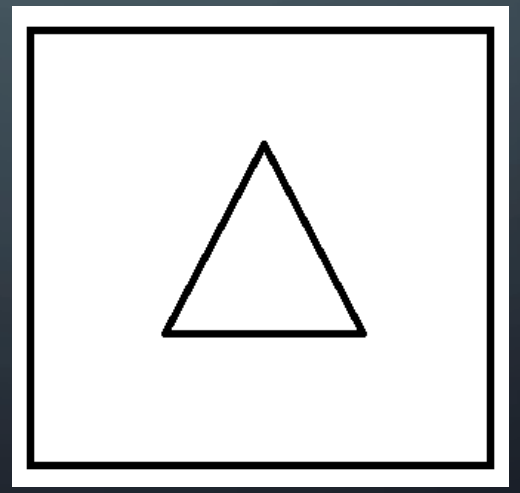
- This is a basic example of a visual analogy
- Square inside triangle is to triangle inside square as circle inside square is too…
  - Multiple choice
- To us, the answer is obvious as choice B

VISION

- Feed in images individually
- This reduces a need to locate different groups in a larger image
- More readable







WRITTEN IN PROLOG

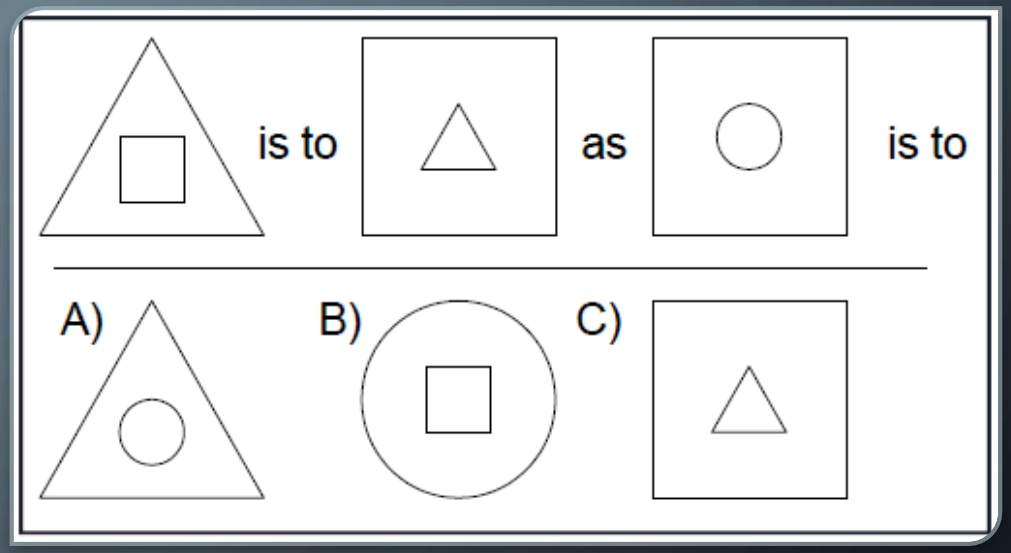
- The Analogy image can be converted into a Prolog text.
- The image becomes a labeled question to be queried
  - Query is analogy_open for open ended analogy
  - Query analogy_multi for multiple choice

```
?- analogy_open(q1r, Answer).
```

```
question(q1i,
         small sized square inside large sized triangle is_to
         small sized triangle inside large sized square as
         small sized circle inside large sized square is_to
         [small sized circle inside large sized triangle,
         small sized square inside large sized circle,
         small sized triangle inside large sized square]).
```

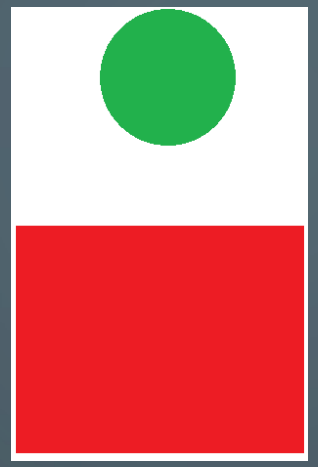
ANALOGY_MULTI

- Original example:
  - 3 multiple choice answers
  - The answer must be one of the 3 choices
  - Or else nothing is returned

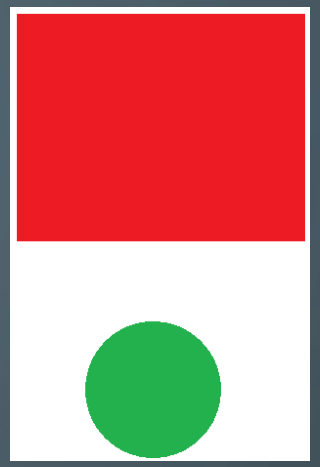


ANALOGY_OPEN

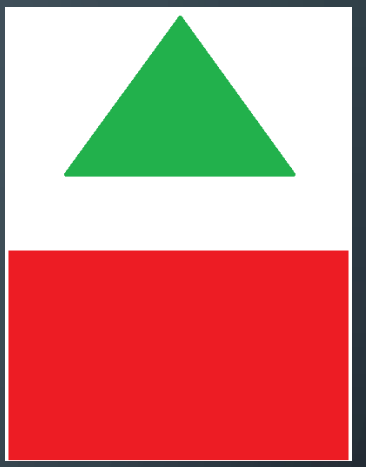
- Analogy open
  - Prolog returns all solutions it finds
  - (usually only 2)



Is to



As



Is to:

A `relation(sized(small, circle), below, sized(large , triangle))`
B `relation(sized(large, triangle), below, sized(small, rectangle))`
C `relation(sized(small, triangle), above, sized(large, rectangle))`

```
?- analogy_open(q1r, Answer)
relation(sized(large, rectangle), aboveR, sized(small, triangle))
relation(sized(large, rectangle), aboveR, sized(small, triangle))
relation(sized(small, triangle), belowR, sized(large, rectangle))
relation(sized(small, triangle), belowR, sized(large, rectangle))
```

WHAT'S IT DOING?

It uses this transformation function

- It's looking for what changed
- Specifically looking for opposites
  - Large/small
  - Above/below
  - Etc.

```
transform(F1 is_to F2 as F3 is_to Solution) :-
    relation(sized(S1old, Shape1), Rold, sized(S2old, Shape2)) = F1,
    relation(sized(S1new, Shape1), Rnew, sized(S2new, Shape2)) = F2,
    relation(sized(S1old, Shape3), RRold, sized(S2old, Shape4)) = F3,
    relation(sized(S1new, Shape3), RRnew, sized(S2new, Shape4)) = Solution,
    value(Rnew, V1),
    value(Rold, V2),
    value(RRnew, VV1),
    value(RRold, VV2),
    VV1 is VV2 + V1 - V2.
```

CONCLUSION

- Prolog can be used to take in visual data
- With better object recognition we can get more data
- Still want to do the search problem in the ECE basement